

BAD :
A BASIC LANGUAGE COMPILER
FOR THE TMS 320 C31 DSK

F. Auger
GE44-IUT de Saint Nazaire
CRTT, Bd de l'Université, BP 406,
F-44602 Saint Nazaire cedex, France

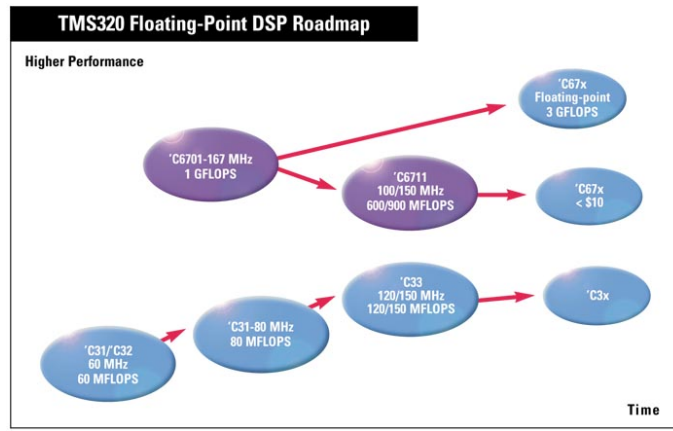
e-mail: `auger@ge44.univ-nantes.fr`



related web site :

`http://crttsn.univ-nantes.fr/~auger/bad`

The Tms 320 c 31 floating point DSP

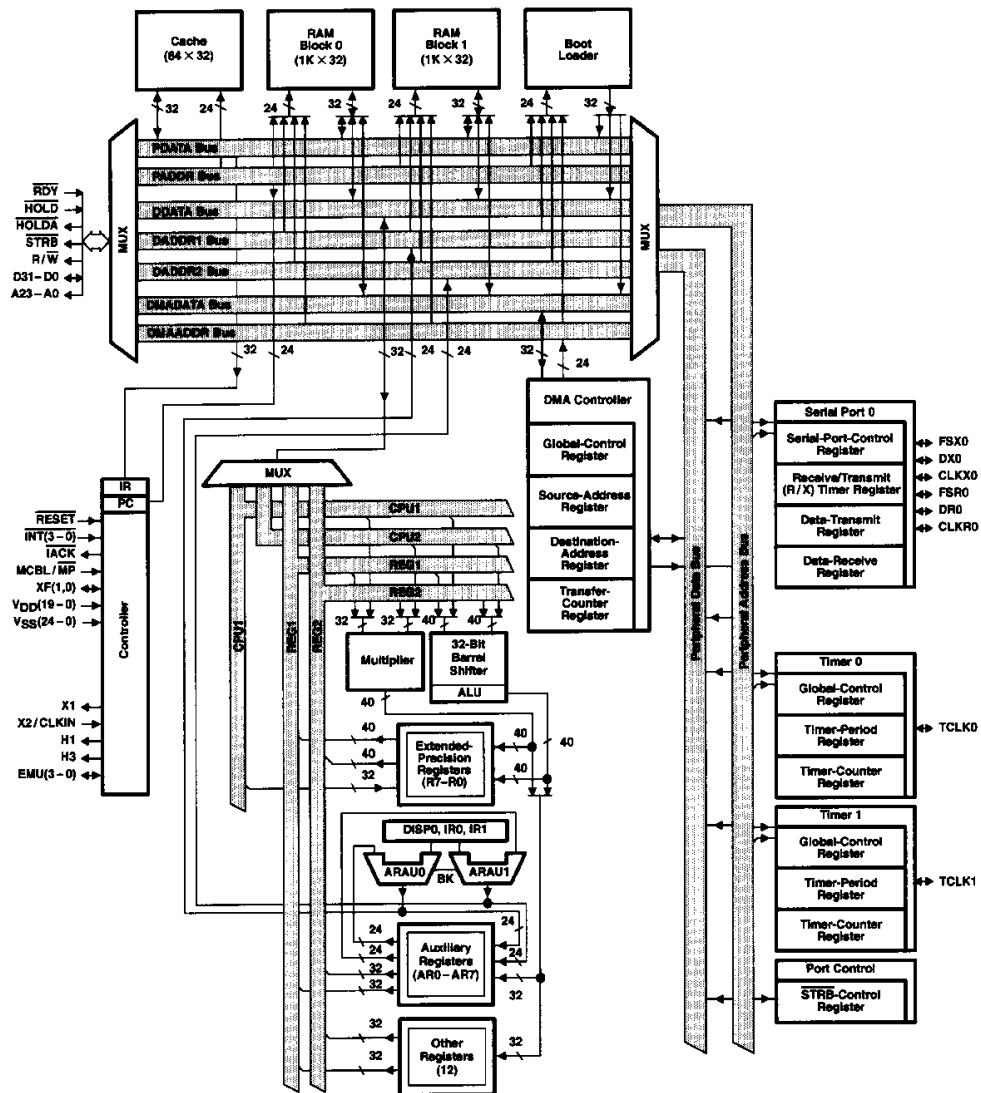


- 32-bit CPU
- 16-bit and 32-bit integer operations
- 32-bit **floating point** operations
- **parallel** arithmetic and multiplier execution
- **50** MFLOPS, **40** MIPS
- **eight** extended precision registers
- 8 auxiliary registers
- **zero-overhead** loops with single-cycle branches
- 2 1K×32-bit **on-chip** RAM blocks
- 1 serial port
- 2 32-bit timers
- 1 DMA channel

The Tms 320 c 31

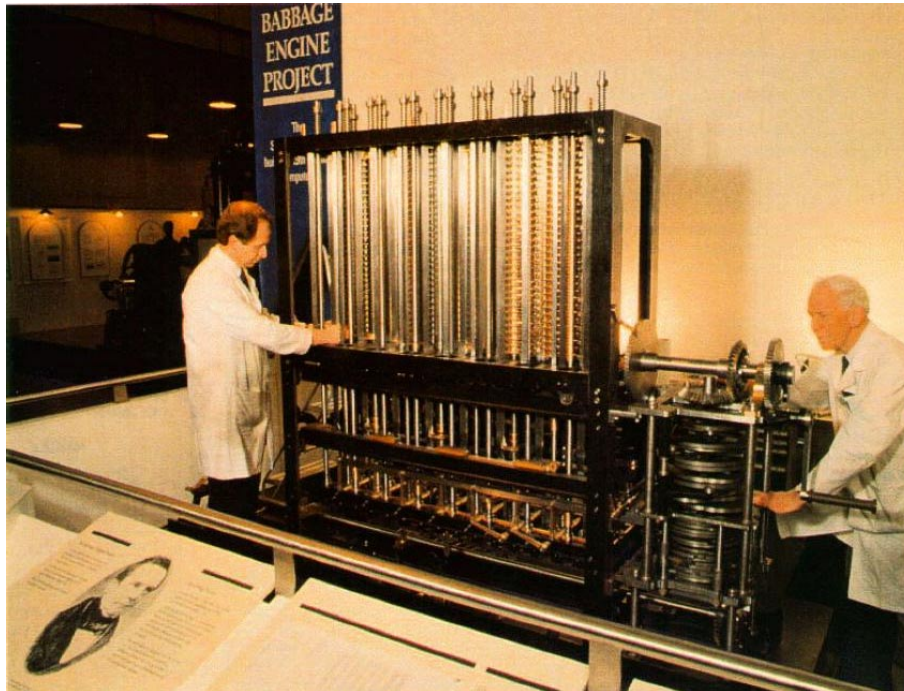


functional block diagram

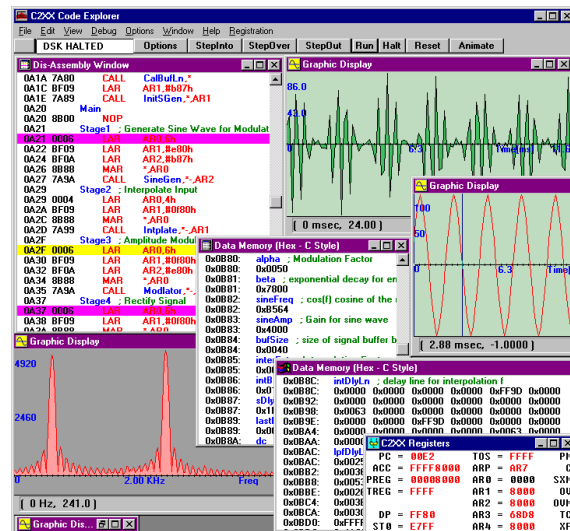
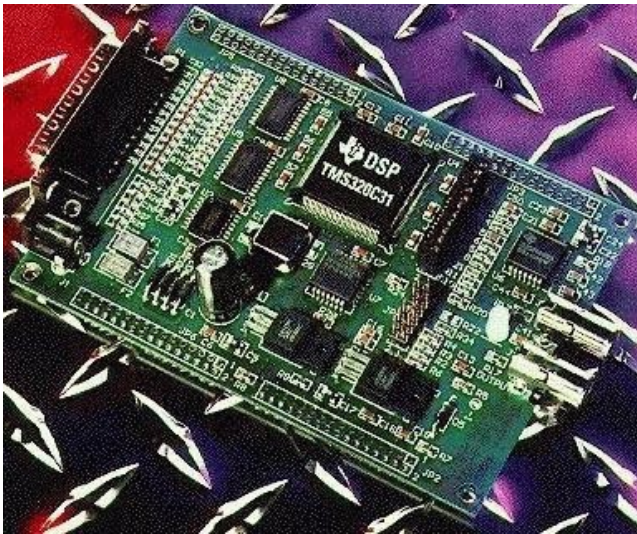


The Tms 320 c 31

slightly smaller than Babbage's difference engine



The C31 Dsk



- low-cost and easy to use development platform
- expandable thanks to an I/O expansion bus
- TLC32040 Analog Interface Circuit for audio applications
- interfaces to a host PC through the parallel printer port
- user-friendly development environment

AVAILABLE PROGRAMMING LANGUAGES I

The C3x assembler

- short and efficient code
- requires knowledge of the DSP structure
- requires knowledge of the DSP instruction set

looks like this :

```
ldi    ST, R0                ; save status register
and    ~2000h, ST            ; temporarily disable interrupts
pop    AR0                   ; bump stack pointer
pop    AR0                   ; get A address
pop    AR1                   ; get B address
addi   3, SP                 ; restore SP
ldi    R0, ST                ; restore status register

mpyf3  *AR0++, *AR1++, R0    ; a[0] * b[0] -> R0
mpyf3  *AR0++, *AR1++, R1    ; a[1] * b[1] -> R1
mpyf3  *AR0, *AR1, R0        ; a[2] * b[2] -> R2
|| addf3 R0, R1, R3          ; (a[0]*b[0]) + (a[1]*b[1]) -> R3
addf   R3, R0                ; complete dot product in R0

rets                                ; the end
```

- “Conventional DSP processors tend to have highly specialized, complicated and irregular instruction sets. This complicates the task of creating efficient assembly language software.”

J. Eyre, J. Bier, “The evolution of DSP processors,” IEEE Signal Processing magazine, vol 17, No 2, march 2000.

AVAILABLE PROGRAMMING LANGUAGES II

The C language

- easy to read and to modify
- But how to use interrupts ?
- How to handle bits and bytes ?



Did **Dennis M. Ritchie** know DSP's when he designed the C language ?

- “Most widely used high level languages, such as C, are **not well suited** for describing typical DSP algorithms.”
- “Conventional DSP architectures are **difficult for compilers** to use effectively.”

J. Eyre, J. Bier, “The evolution of DSP processors,” IEEE Signal Processing magazine, vol 17, No 2, march 2000.

AVAILABLE PROGRAMMING LANGUAGES III

The proposed Basic language



Thomas E. Kurz, John G. Kemeny,
inventors of the BASIC language

- freely available at
<http://crttsn.univ-nantes.fr/~auger/bad>
- designed for students and application engineers
- can be seen as an algebraic assembly language
- case insensitive
- predefined variables allow a direct use of 4 internal registers
- generates an easy to understand source code, which can be locally hand-optimized



STANDARD PROGRAM IN BAD

- Program header

`dspbasic(c31dsk)`
declares the target

- constants and variables declarations

`const length 10`
`var y float`
`var j int`

- program instructions

runtime math and logical expressions
label definitions
unconditional jumps
conditional statements
subroutine calls
allowed use of software and hardware interrupts

looks like this :

```
dspbasic c31dsk
' test program for the basic dsp compiler
' F. Auger, june 1999

var SETSP      int 0E970300h      ' serial line configuration
var AICSETSRDA int 1,5,0x1f,0x19  ' TPR TA TB (pol_it*128 + CRW)
var twaitsrda  int 0              ' useless srda (no parameters)
var AICIOSRDA  int 0,0            ' CNA CAN

' define the variables of the main program

var SineVal    int 0, 707, 1000, 707, 0, 0, -1000, 0
var indice     int 0
var length     int 8

:main
  aicsetsrda(3) |= 128 ' Interrupt mode configuration
  gosub aicset

:ITLoop
  idle
  goto ITLoop

:xint0
  aiciosrda(0)=SineVal(indice)
  gosub aicio
  indice+=1
  indice&=7
  reti
```

CONCLUSION

For this environment to be broadly used, its library of available signal processing tools must be expanded

This compiler can be fitted to other DSP targets